Unix as the Interface of an On-line Text Corpus

Leho PALDRE
Tartu Calvary Church
Kuldnoka 15
Tartu, Estonia, 50404
lpaldre@psych.ut.ee

Heiki-Jaan KAALEP
Dept. of Estonian and Finno-Ugric Linguistics
University of Tartu
Tiigi 78-206
Tartu, Estonia, 50410
hkaalep@psych.ut.ee

Tarmo VAINO
Dept. of Estonian and Finno-Ugric Linguistics
University of Tartu
Tiigi 78-206
Tartu, Estonia, 50410
tvaino@psych.ut.ee

Abstract

The article describes an on-line text corpus interface – a Unix command window. A corpus should be usable via Internet, to allow maximum access to it. When devising our corpus query interface, we wanted to avoid complications resulting from "making a special corpus interface for linguists" which would most likely mean a slightly unconventional syntax, partly lengthy and partly missing documentation, and buggy routines. Instead, we decided to provide the users a Unix command line window, in addition to a simple *grep*-like query for less complicated search.

Introduction

The increasing sizes of language corpora and higher demands on their processing leads to finding new ways to make corpora easily accessible. It is not conceivable any more that a linguist would turn to a computer specialist every time he has a more complicated task than just a simple lookup of a word.

This has motivated us to making corpora publicly available via Internet and providing them with intelligent processing tools. The Unix query interface we describe has been implemented on the Corpus of Estonian Written Language that is being built by the University of Tartu.

1 Description of interface

The Corpus of Written Estonian (CWE) is internally represented as a set of lines: one line is one sentence. We offer the user a Unix -bin/unix_sj_en.cgi) so that relevant Unix commands can be entered for queries over the whole corpus or parts of it.

Commands are run on complete sentences (i.e. lines) and can be used singularly or piped into more complex queries. How powerful the interface is depends very much on one's knowledge of Unix. In principle there are endless combinations of commands.

1.1 From *grep* to full Unix

This kind of interface was prompted by the need of linguists to have a friendly and powerful access to the corpus. Our first attempt was to provide a query that uses *grep* at its background for simple concordances. It can be run both over texts tagged up to the level of sentences and over morphologically tagged texts.[1] Similarly one can use regular expressions and/or search for morphological tags in CWE. This first solution was sufficient for simple concordances but did not allow further refinements of results nor more complicated queries.

We thought it wise not to invent any further *ad hoc* query language of our own that would then need thorough documentation and differ from conventional languages mostly by annoying modifications in syntax.

Traditional Unix facilities seemed a reasonable solution. Unix commands have become standard, are widely used in computational linguistics and they are sufficient

[1] The morphological analyser was developed by H.-J. Kaalep and T. Vaino. It can be used freely from .

for most corpus queries, therefore no additional features or pre-processing is necessary.

For the sake of security only a choice of Unix commands has been made available for corpus research: *cut, egrep, grep, head, join, paste, rev, sed, sort, tail, tr, uniq, wc*. Each of them is linked to an appropriate man-page so there is no need for extra documentation. We have set a constraint that no output can be saved to the WWW server. This should guarantee against abuses of the interface.

1.2 Comparison with other interfaces

Our *grep*-like query is quite similar to several other interfaces. For example IMS Corpus Query Processor developed by University of Stuttgart[2] allows special attributes and uses regular expressions in queries. British National Corpus[3] offers more limited search possibilities. The Canadian TransSearch[4] is even simpler but suggests parallel context in another language as output. We have not dealt with queries on parallel-translated texts, as CWE is a monolingual corpus.

We have not met Unix implemented as a query language for an on-line text corpus elsewhere. The interface we provide makes no limitations on the amount of corpus processed and downloaded. This way it differs from several other publicly available corpora. Our interface includes most of the facilities of the query languages mentioned above in addition to the extra possibilities.

At present it runs on a corpus of several million words that actually consists of subcorpora of texts from different decades of this century. The contemporary texts have been morphologically tagged. In principle the power of our interface depends on how much the corpus has been tagged. So the additional documentation we have to provide is about how the text has been tagged. The routines themselves are standard Unix.

[2] -stuttgart.de/projekte/CorpusWorkbench/

[3]

[4] -rali.iro.umontreal.ca/TransSearch/TS-simple-uen.cgi

2 Actual usage

Our corpus interface has mainly been a tool for Estonian language research for university students and researchers. Mostly the *grep*-like concordance page is used for simple queries in order to get example sentences for various purposes. This can be concluded by the fact that the number of hits per visitor to this page is relatively small. It requires no prior knowledge of any query language, is easy to handle and one gets the required result without much experimenting.

Usage of our Unix query page has been surprisingly active according to statistics: nearly 3800 hits within five last months. It is only 1000 hits less than the simpler search page. It has proved very useful for scientists with specific tasks who carry out research on actual language usage, compile mono- and multilingual dictionaries and explore language change.[5]

Majority of users comes from within Estonia. Some hits originate from Finland that speaks a language, belonging to the same family with Estonian. The number of other visitors is few and can be neglected.[6]

Most of the users, however, have presumably been students of our own university who have needed it for their assignments. Almost 3300 hits to the Unix page (88%) date from the month it was taught as part of a class at the university.

Our experience with students has been that without having any prior knowledge of Unix one can acquire basic skills and use Unix as a query language after four hours of hands-on tutoring. In reality this has meant 2-3 practical classes within a computer skills course oriented to students of linguistics who had had only a general experience of using a computer. Explaining regular expressions took one third of the time. Creating a frequency list based on CWE was the final task of one course.[7] Most students were able to complete it.

[5] CWE has subcorpora with representative texts from most decades of this century.

[6] One reason for this is, of course, that the English version of our Unix query page was added much later than the Estonian one.

[7] This assignment can be found at

The main complaint that we have heard as feedback is that Unix is too complicated to be used by a usual linguist. We think that any interface needs a certain amount of instruction in order to enter as complex queries as can be done with Unix commands. It needs further testing whether learning Unix is slower or faster than some other query language.

Another feedback has been that in case of Unix queries one easily gets an awful lot of sentences that takes time to download. We have advised our students to start with a *head*-command while experimenting.

3 Discussion

3.1 Matters to concern about

There are several issues with our interface that might cause problems to users or builders. Although they do not directly concern the tool itself, they are to be taken into account when implementing and developing such an interface.

Processor load and slow connections

When corpora grow larger we come to the question of processor load and slow connections.

We have met some occasional error messages «Document contains no data» when 15 students have tried to perform the same query simultaneously during a computer class. But in fact, we are not concerned with the computational power and bandwidth: the hardware is developing so quickly to keep in pace with video and sound processing and transporting, that text processing and transporting are really no problem.

It's hardly possible to compare the speed of our searches to similar corpus query tools because it's very dependant on external circumstances like Internet bandwidth. For the same reason we have not thought it reasonable to provide the user with run-time feedback about how processing is going on and how long the current process is likely to take. It is, however, possible to cancel the job and return to the previous stage without disturbing the system if the process takes too much time or when an error is discovered.

It is problematic to attempt to set some standard limit to the amount of output in the case of Unix queries. In case of simple searches the output could well be limited to a certain number of lines (sentences), and this has been implemented on the *grep*-query interface. But once we add *sort* and *uniq* facilities, this kind of limitation makes no sense.

Copyright

As a prerequisite for a corpus interface that in principle allows one to download the whole corpus, the copyright restrictions on the texts have to be minimal. One way to achieve this is by creating the corpus by collecting excerpts, not whole texts, like in the LOB[8] and Brown[9] corpora. This is what we have done also.

Copyright problem concerns chiefly the output. In principal, this could be limited by controlling the access rights of the users (independently of the interface itself), or by checking that the output differs enough from the original, copyright-protected chunk of text. Both methods would need considerable effort from the corpus owner, so we haven't implemented them.

Unfamiliarity with Unix

Another prerequisite for using this corpus interface is some amount of knowledge of Unix commands. We believe, however, that with the help of example queries this obstacle can be overcome. Unix users can have very different levels of command. The most frequent queries can easily be learned. Using *copy & paste*, they can easily be reused also.

Currently the user gets no original Unix error messages if he has made a mistake in the command line. The only error message we have implemented prompts the user if he has used an unknown (or un-allowed) command.

Internal format

To formulate queries, the user should be familiar with the internal format of the corpus. At present it is sentence-per-line. In the morphologically annotated part, the (often ambiguous) analyses are on the same line with the words, delimited by special symbols; the whole sentence is one single line. As a further development the internal format of the annotated part can be made

[8] Cf. lob-cont.html

[9]

simpler by disambiguating the analyses, so that the user can take more advantage of it. This in turn depends on the development of tools for this purpose.

We have used HTML characters for special symbols in order not to make queries dependent on any particular code page. This makes entering the text somewhat clumsier. A solution would be to let the user choose how to enter these characters.

We have not made any effort to link the corpus with some dictionary, or the texts with parallel texts.

3.2 Matters to be content with

We believe that our interface has several advantages due to the fact that the query language is Unix.

Flexible interface

With Unix as interface one can make very complicated queries, if necessary. This flexibility is confined to the fact that one cannot save anything to a file, or combine two or more files as input. This would make *join* command much more powerful.[10]

The possibility of piping commands has proved very useful in refining output when the initial output includes systematically superfluous information. Finally, the output can be personally designed so that important items are easily recognisable (turning keywords into bold, highlighting them with special symbols etc.). And there is always the possibility to save the output to one's own computer and continue analysing with other tools.

Extendable interface

The number of available commands does not have to be limited to our choice. The corpus interface provider could add new tools that the users can exploit in a command pipe. In linguistic research a morphological analyser and a disambiguator would be marvellous tools to use. Those would then need extra documentation, of course.

[10] In principle one could conceive of making a default dictionary or word list available that can be used as a second input file for *join* in order to compare corpus information with some other data.

Unix: a well-known standard

We have not modified Unix in any way. The query language (Unix commands in a pipe) is a well-known standard – that has not changed since it was developed in 1976 – with ample documentation. In practice this can be one of the most relieving arguments for setting up such an interface: no need to create and update user manuals.

The interface is easy to implement

It needs no deep skill to put up an interface like ours. A practical hint is to use GNU commands instead of the ordinary ones. E.g. sentences in a corpus can be extremely long, and unlike some commercial versions, GNU *sed* and *grep* can handle lines of infinite length.

Query languages usually assume some kind of advanced internal structure of the corpus. This can be so and is an advantage in case of Unix, but it is not inevitable. Unix tools are powerful enough on plain untagged text. Further tagging adds linguistic relevance to queries but the engine remains the same.

Conclusion

Creating a Unix-based interface for an online corpus has proved very useful for linguistic research. Tasks that have so far required special access and special knowledge can now be solved nearly by anyone.

Although our interface needs more learning than a simple lookup query it has power that the simpler query languages lack. Both kinds of interface are necessary and therefore we have retained the simpler search in addition to the more advanced one.

Acknowledgements

Unix interface for CWE would not have been possible without the funding from the Open Estonian Foundation and the readiness of the Estonian Educational and Research Network (EENET) to provide its server.

The members of the research group of Computational Linguistics of the University of Tartu have been most helpful in using the interface and pointing out initial bugs.

Special thanks to Tomaž Erjavec for helpful links to other corpora on his web pages.